An Adaptable Robot Vision System Performing Manipulation Actions with Flexible Objects

Bodenhagen, Leon; Fugl, Andreas Rune; Jordt, Andreas; Willatzen, Morten; Aulkjær Andersen, Knud; Mølbach Olsen, Martin; Koch, Reinhard; Petersen, Henrik Gordon; Krüger, Norbert

Published in:
I E E E Transactions on Automation Science and Engineering

DOI:
10.1109/TASE.2014.2320157

Publication date:
2014

Document Version
Early version, also known as pre-print

An Adaptable Robot Vision System Performing Manipulation Actions with Flexible Objects

Leon Bodenhagen, Andreas R. Fugl, Andreas Jordt, Morten Willatzen, Knud A. Andersen, Martin M. Olsen, Reinhard Koch, Henrik G. Petersen, and Norbert Krüger

***Abstract*—This paper describes an adaptable system which is able to perform manipulation operations (such as Peg-in-Hole or Laying-Down actions) with flexible objects. As such objects easily change their shape significantly during the execution of an action, traditional strategies, e.g. for solve path-planning problems, are often not applicable. It is therefore required to integrate visual tracking and shape reconstruction with a physical modeling of the materials and their deformations as well as action learning techniques. All these different sub-modules have been integrated into a demonstration platform, operating in real-time. Simulations have been used to bootstrap the learning of optimal actions, which are subsequently improved through real-world executions. To achieve reproducible results, we demonstrate this for casted silicone test objects of regular shape.**

***Note to Practitioners*— The aim of this work was to facilitate the setup of robot-based automation of delicate handling of flexible objects consisting of a uniform material. As examples, we have considered how to optimally maneuver flexible objects through a hole without colliding and how to place flexible objects on a flat surface with minimal introduction of internal stresses in the object. Given the material properties of the object, we have demonstrated in these two applications how the system can be programmed with minimal requirements of human intervention.**

Rather than being an integrated system with the drawbacks in terms of lacking flexibility, our system should be viewed as a library of new technologies that have been proven to work in close to industrial conditions. As a rather basic, but necessary part, we provide a technology for determining the shape of the object when passing on e.g. a conveyor belt prior to being handled. The main technologies applicable for the manipulated objects are: A method for real-time tracking of the flexible objects during manipulation, a method for model-based offline prediction of the static deformation of grasped, flexible objects and finally a method for optimizing specific tasks based on both simulated and real-world executions.

***Index Terms*—Flexible objects, deformation modeling, action learning, 3D-modeling, shape tracking.**

Manuscript received August 20, 2012. This work was co-financed by the INTERREG 4 program Syddanmark-Schleswig-K.E.R.N. by EU funds from the European Regional Development Fund. The presented work has also received funding from the EU Seventh Framework Programme under grant agreement no. 270273, Xperience.

L. Bodenhagen, A. R. Fugl, H. G. Petersen, and N. Krüger are with the Maersk McKinney Moller Institute, University of Southern Denmark, Odense, Denmark (e-mail: {lebo, arf, hgp, norbert}@mmmi.sdu.dk).

A. Jordt and R. Koch are with Institute of Computer Science, Christian-Albrechts-Universität, Kiel, Germany (e-mail: {jordt, rk}@mip.informatik.uni-kiel.de).

K. A. Andersen and M. M. Olsen are with the Danish Technological Institute, Denmark (e-mail: mmo@dti.dk).

M. Willatzen is with the Mads Clausen Institute, University of Southern Denmark, Sønderborg, Denmark (e-mail: willatzen@mci.sdu.dk) and the Department of Photonics Engineering, Technical University of Denmark, Ørsteds Plads, Building 345v, DK-2800 Kgs. Lyngby, Denmark.

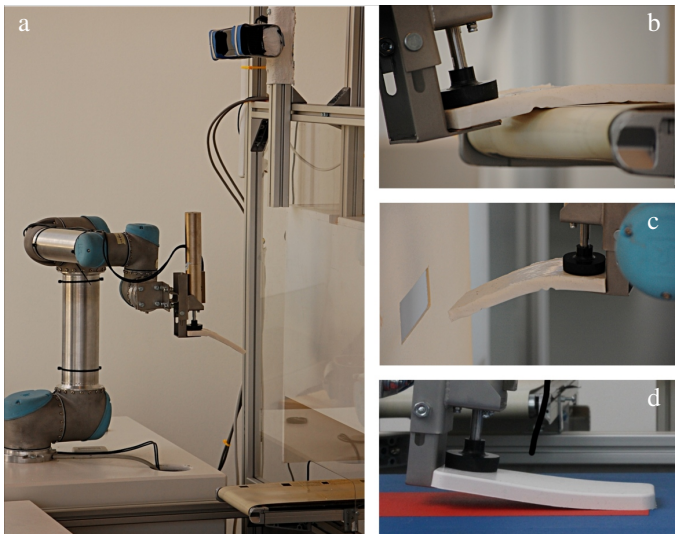


Fig. 1: a) Whole set-up. b) Grasping process. c) Peg-in-Hole action d) Laying-Down action.

I. Introduction

In the last decade, visual sensors have become a more and more important factor in industrial production as they allow for the handling of uncertainties of object poses in the manipulation process. For static objects, this has led for example to pick and place robot systems that can operate in reasonably constrained scenarios. Bin picking systems that are able to operate in more complex contexts, such as major clutter in a bin of randomly distributed rigid objects, have now entered the market [1]. Recently, work was performed on learning and fine-tuning of manipulation actions by means of experience gathered in simulation as well as in the real production context [2]. This provides additional means for increasing the flexibility in production. In addition it reduces the need to design highly tuned systems with close to 100% performance which usually requires a large amount of engineering expertise which leads to high setup costs of a robot solution.

The manipulation of flexible objects in a dynamic and unconstrained situation poses a number of additional challenges on robotics when compared to the manipulation of rigid objects. First, the configurations of rather complex object shapes need to be sensed in 'real–time', i.e. computed fast enough to allow for a robot action. Second, properties of the material relevant for the modeling of the deformation process must be sensed visually or haptically. Third, the actual deformation of the object under external forces is difficult to

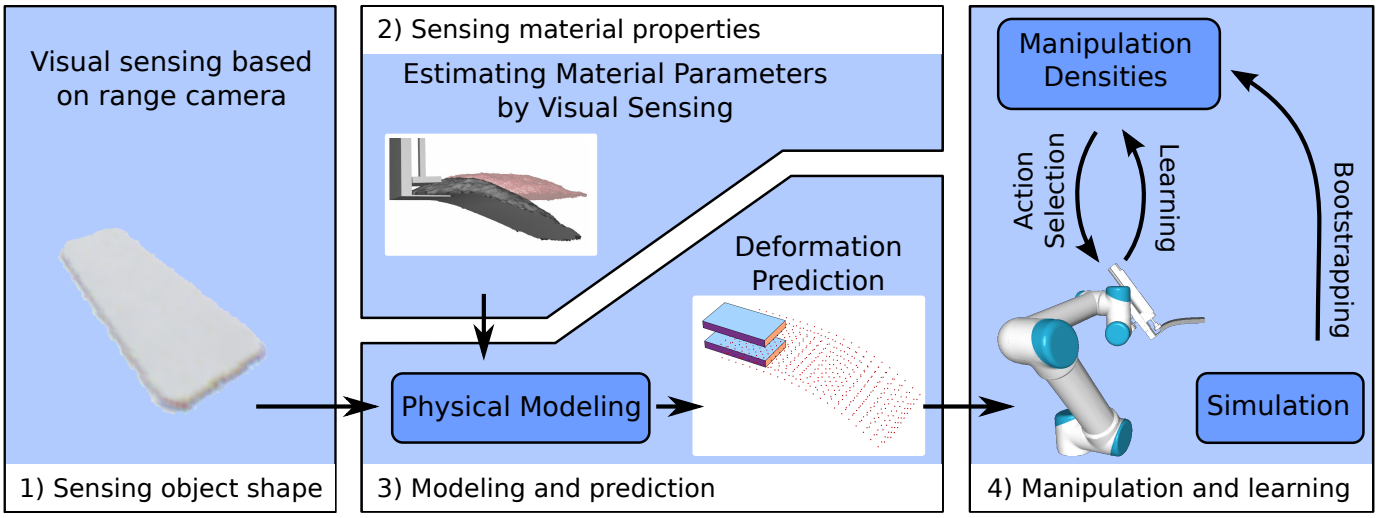


Fig. 2: Overview of the manipulation approach and the four modules.

model precisely and depends on the actual properties of the material. Fourth, optimal manipulation trajectories need to be computed and performed in a real system. In this context, it is unlikely that all parameters of the system can be estimated by analytic modeling only and hence it will be important to integrate some kind of adaptability in such a system.

In this paper, we describe a system which tackles all four problems mentioned above to perform grasping and two manipulation tasks (a Peg-in-Hole insertion action and a Laying-Down operation) with flexible objects. Fig. 1 shows the physical set-up along with the grasping and manipulation tasks.

The different steps of the process are illustrated on Fig. 2 and addressed in the following way:

1) Sensing of Object Shape: The acquisition process is split into two separate problems: (1) Capturing the 3D shape of the object in one stage and (2) tracking its deformation and movement during the interaction with the robot in a second stage. The 3D data is first acquired while the object remains static on the conveyor (see Fig. 3a) providing an optimal environment to retrieve the object shape accurately. In a second stage, the object is tracked while it is deformed and manipulated, using the already known shape provided by the previous stage as a reference object.

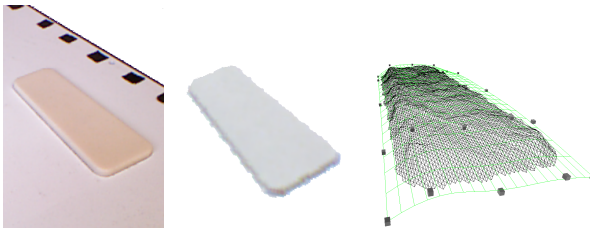
2) Sensing of Material Properties: Material properties are estimated using 3D shape tracking and physical modeling. A search for the best fit to material properties to an underlying physics model is performed, while fitting deformation parameters and object pose to visual data.

3) Physical Modeling and Prediction: We use a mathematical representation of flexible objects based on the general elasticity equations of an isotropic medium. Depending on the accuracy required for a particular part of the system, we use either linear beam models [3] or employ more complex models incorporating non-linear geometric strains and volumetric effects [3], [4].

4) Manipulation and Learning: Learning is applied in two situations: for computing close to optimal Peg-in-Hole action (see Fig. 1c) and the Laying-Down action trajectories first in simulation and then in a fine-tuning step in the real world system (see Fig. 1d). The learning tasks are addressed by extensions of the concept of Grasp Densities [5] which is based on Kernel Density Estimation (KDE) [6]. The densities are bootstrapped using simulated experiments as subsequently refined based on real experiments.

In this paper, we will show results for the different sub-modules as well as the overall system. Finding stable solutions for the first three problems and a successful integration into an embodied system opens possibilities in a number of application fields from industrial robotics (e.g., food production, manufacturing, etc.) to service robotics (e.g., feeding robots, helpers in kitchens, etc.). Earlier attempts have been made to create robot systems for the handling of flexible objects (see, e.g., [7]–[9] and section II). However, by specializing the underlying physical model they were often restricted to a narrow class of objects, e.g. for handling cables [10], or handling sheets of paper [11]. We envision the ability to model and handle a much broader range of objects by the use of 3D vision, general continuum elasticity and learning. By that, this paper presents an example of a manipulation system in which real-time vision, modeling and learning first take place in simulation and then are combined in the real application context. This gives new perspectives for establishing manipulation systems in production by reducing engineering costs in the design process of the automated solution by the integrative use of sensing, modeling and learning.

The structure of the paper is as follows: In section II, the current state of the art is outlined with respect to the individual modules. Detailed descriptions of the modules are given in section III and results as well as an outline of the overall system is provided in section IV. Prior versions of the sub-modules of the system – which are presented in a very condensed way in this paper – have been published at conferences in [12]–[17]. We refer in this paper to these works



(a) (b) (c)

Fig. 3: From left to right: Photo of the real object during the scan; The colored 3D scan of the object; 3D data from the Kinect camera showing the object; The object mesh (black) and the deformation surface (green).

for more detailed descriptions of the sub-modules. In contrast to [12]–[17], the main topic of this paper is the process as a whole and a proper quantification of this process. Also it turned out that some of the submodules need to be further extended to integrate them properly in the overall process. Hence in this journal paper, we present the final, mature system.

II. State of the Art and Own Contributions

Creating a system that is able to handle flexible objects requires the integration of a variety of different disciplines such as computer vision, solid mechanics and machine learning. In the following an overview of the relevant related work as well as the contributions of this paper is provided separately for the four challenges outlined above.

A. Sensing of Object Shape

Acquiring a 3D shape of a static object has been subject to research for a long time [18]. For the task at hand, we generate a 3D model from the depth image of a Kinect that is mounted above the conveyor.

Tracking the 3D deformation of objects is a task far more complex. The research performed in this area has been done with a wide range of applied hardware and algorithms. Incorporated sensors range from multi-projector [19] and multi-camera systems [20] [21], active range cameras [14] down to stereo [22] and even monocular camera systems [23], from which the deformation has to be computed. For the task at hand, we will not be able to observe the object from all sides because of the interaction with the robot. Hence we will stick to an approach similar to [12], an approach for tracking flexible objects directly in the input data that does not require any form of data preprocessing (as e.g. feature detection [23] or background subtraction [24]) or the utilization of markers [25]. In our test setup, we use a Microsoft Kinect to acquire color and depth information. In a first step, a single frame is used to generate a static 3D shape (Fig. 3b). In a second step, the deformation of the model is tracked using non-uniform rational B-splines (NURBS) [26] (Fig. 3c) function in the generic case [12] or a simple bending deformation in the special case to track the object deformation as it is grasped by a robot gripper on one end [16]. The deformation tracking uses an efficient optimization scheme to search for deformation parameters which fit the range- and color images of the Kinect best.

B. Sensing of Material Properties

Elastic moduli, such as Young's modulus and Poisson's ratio, are central to describe the elastic properties of an object. E.g. Young's modulus can be seen as the stiffness of the object, while Poisson's ratio characterizes its compressiblity. For any simulation it is necessary to have the elastic moduli reliably determined. There is a wide range of methods for determining the elastic moduli for a deformable object, but not all are applicable in a robotics workcell. The most direct way to measure the elastic moduli of an isotropic material is to homogeneously deform a regularly shaped sample and measure the resulting force. Given careful control of boundary conditions, this can be done very accurately [27]. Similarly indentation tests are performed by pressing a small rod onto the elastic surface and from this infer the stress-strain relationship. Indentation tests have been used successfully to measure soft elastomeric materials [28]. Ultrasound imaging was used in [29], however, large relative errors in estimating Young's modulus were present.

Few authors have presented work directly relating to robotics workcells. Howard and Bekey [30] proposed how to learn parameters for a damped-spring model in order to calculate the minimum lifting force required to manipulate deformable objects. In [31], Frank et al. used a robot manipulator equipped with a force-torque sensor and a stereo camera to observe the deformation of flexible objects. By obtaining the deformation by vision along with the force reading from the sensor when using an indenter, they used a volumetric model to search for Young's modulus and Poisson's ratio. However, no validation of the elastic moduli was made. In our system, we use a similar approach, in that we fit an underlying deformation model to the observed data. Instead of a classical regression setup, however, we extend the tracking concept applied to address the first challenge (see section III-A) with a simultaneous search for the material properties. Observing the object deforming under gravity, we simultaneously estimate material properties and pose.

C. Physical Modeling and Prediction

The underlying elasticity equations for a general elastic material are well-known and solutions exist for many special cases [3], [32]. However, a complete and numerically tractable physical model of a flexible object subject to a comprehensive set of relevant boundary conditions (as often would be required in the context of manipulation operation) is not available in the literature today.

Much work has been done to tailor mass-spring models to suit the physical parameters of real material. In [33] mass-spring models are used to mimic elasticity in cloth and thread handling using robots and in performing real-time surgery simulation. The approach however was manually tuned for both spring topology and spring constants and no track of

modeling errors was presented. In [8] a system for the characteristics of grasping flexible objects is outlined based on mass-spring models. The model was intended to supply estimates for the required grasping force, to reliably manipulate an object. However, not many simulation results were presented, and no measure on modeling errors has been provided. The same approach to employ mass-spring models was used in [34]. They showed realistic contact behaviour for pure compression, but admitted problems with handling shear strain due to their spring topology.

In our system we use the general equations of elasticity and reduce dimensionality where applicable. For low accuracy tasks and moderate deformations we employ fast, linear models and for high-accuracy tasks involving larger deformations we employ geometrical non-linearity. This leads to a richer and more complete spatio-temporal description of the elastic state as compared to mass-spring models.

D. Manipulation and Learning

An analytic definition of optimal actions quickly becomes intractable as the space of actions often is high-dimensional. Also the actual actions can only be modelled to a certain degree due to required approximation in the actual modeling and sensorial uncertainty. Hence, in addition to modeling, learning can be used to identify robust actions. Grasp poses can for instance be learned by exploring the space of grasp affordances of an object [5], [35]. More generic types of actions can be learned by demonstration [36], [37]. However, these approaches do not allow for a generalization of actions across different objects. Although it could be considered that it is possible to learn an action for a specific flexible object using existing methods, one has to remember that a non-rigid object can behave fundamentally differently depending on the current external forces acting on the object. Therefore not only the action and the object as such need to be addressed in the learning schema, but also the current state of the object. To account for these complexities, we apply a learning approach based on Kernel Density Estimation (KDE), [5], [6] estimating the distribution of successful actions and characteristics of the object in the current situation.

To our knowledge, little work addressing the manipulation of non-rigid objects has been done in an industrial domain so far (see also [38]) and there exist only few adaptable systems in industrial settings. Both facts might be tightly connected since the complexities involved in the manipulation of flexible objects do not allow a full modeling and hence require some kind of learning to be applicable in an industrial context. Therefore, we think that the exploration of the potential of such adaptable systems is an important step towards more intelligent production systems in future production.

III. Description of the four Modules of the System

The overall architecture of our system is sketched in Fig. 2. Starting with the visual extraction process as outlined in section III-A (see Fig. 2-1 *'Sensing object shape'*) and the estimation process as outlined in section III-B (see Fig. 2-2 *'Sensing material properties'*), the physical modeling of the material is performed which leads to the prediction of the state of the flexible object at the moment of manipulation (see section III-C and Fig. 2-3 *'Modeling and prediction'*). This allows then also for the computation and evaluation of action trajectories in simulation which can be utilized to learn promising actions (see section III-D and Fig. 2-4 *'Manipulation and learning'*). Appropriate actions are then executed and evaluated on the real platform and are then used for further fine-tuning. This is described in section IV-B and IV-C.

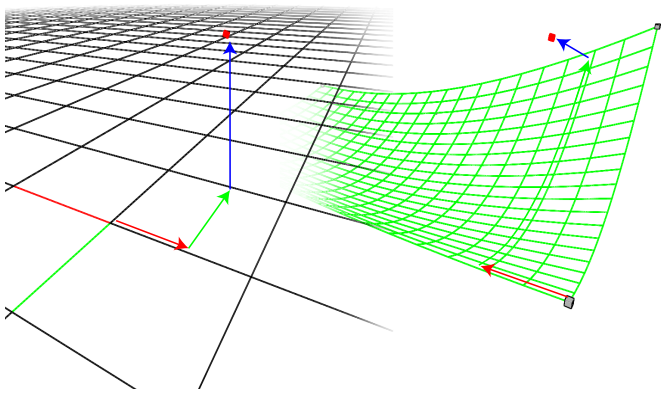
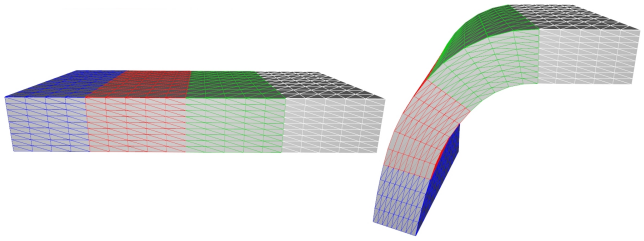


Fig. 4: The transition from (x, y, z) (left) to (u, w, o) (right). A vertex is described by the parameters (u, w) for the closest nearest point on the NURBS surface and an offset w.

A. Sensing of Object Shape

There are multiple ways to describe the deformation of a 3D shape. The most intuitive and the most common way for representing shapes given as triangle meshes is to simply displace the vertices of a mesh in 3D space. It is widely used in deformation tracking (e.g. [21]–[23]). But as a side effect, this very high dimensional degree of freedom entails ambiguities and under-determined equation systems, so every approach fitting vertex positions directly to the data usually comes with a set of additional side constraints and regularization terms attached to it. For an approach aiming at producing results in real-time, a search space of much lower dimension is required, containing regularization and additional constraints implicitly in the domain reduction of the search space.

In our setup, NURBS surfaces [26] can be used to approximate the object surface very roughly and serve as a deformation function [12]. Its degree of freedom does not accommodate the actual surface shape of the object but its deformation, which leads to a surface function described by only a few control points. The missing high-frequency information on the object's surface is added to the NURBS surface function for each mesh vertex separately with a displacement value. For the special case of an object being held by the robot gripper, a dedicated, even lower dimensional deformation function can be used [16].



(a) (b)

Fig. 5: The object model is divided into zones. Each zone is subject to a constant curvature, allowing to simulate deformations caused by gravity when one end is grasped by the robot.

To register a set of vertices to a NURBS surface, for each vertex the parameter of the NURBS surface point closest to this vertex is saved, along with its offset from the surface. This registration can be seen as a conversion of the vertex coordinates from the world coordinate system (x, y, z) to the NURBS coordinate system (u, w, o), where u and w are the parameters of a NURBS surface point and o is the offset perpendicular to the surface (see Fig. 4). The NURBS surface is only used as a deformation function, not as a surface to match to, only the set of vertices is considered in the following optimization, i.e. the NURBS surface does not have to resemble the object's outline but only needs to enclose it so every (x, y, z) can be expressed as (u, w, o). Also, the (inner) NURBS surface does not necessarily have to approximate the objects surface, as deviations between the NURBS and a vertex are compensated by o.

Given the projection information of the calibrated [39] setup and the object position, the deformation tracking can be expressed as a minimization problem.

Let V be the set of mesh vertices $v_i \in \mathbb{R}^3, i = 0, ..., |V|$ and let C_0 be the set of the initial control points of the NURBS function. To bind the scanned object to the NURBS function, the corresponding parameter $p_i \in [0, 1]^2$ in the parameter domain of the initial NURBS function $\mathcal{N}_{C_0}$ is stored for every vertex v_i of the scanned mesh, along with the displacement information $o_i \in \mathbb{R}$ which is the distance between this vertex and the NURBS surface such that

$$o_i = \|v_i - \mathcal{N}_{C_0}(p_i)\|. \tag{1}$$

If $\bar{\mathcal{N}}$ denotes the surface normal function for $\mathcal{N}$, then each of the original scanned mesh vertices v_i can be expressed as

$$v_i = \mathcal{N}_{C_0}(p_i) + o_i \cdot \bar{\mathcal{N}}_{C_0}(p_i). \tag{2}$$

This way, low resolution deformation can be applied to the high resolution mesh by manipulating the control points C of the NURBS $\mathcal{N}_C$ while keeping the deformation domain small and implicitly smooth.

For gripper based interaction, we are able to reduce the set of generic deformations to deflections typical for this scenario, described by a simple bending transformation [16]. Let $\mathcal{B}$ be this deformation function. The undeformed object is divided into zones as shown in Fig. 5. While leaving the white zone unchanged (as it represents the part of the object that is within the gripper), $\mathcal{B}$ applies a curvature to the object that is constant within each of these colored zones. For a single zone, the corresponding deformation curvature is applied by

$$\mathcal{B} : \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x\cos(\alpha) - (z-r)\sin(\alpha) \\ y \\ (z-r)\cos(\alpha) + x\sin(\alpha) + r \end{pmatrix} \tag{3}$$

where $\alpha = \frac{x}{2\pi(r-z)}$ follows a curvature with radius $r = \frac{x_{size}}{\beta}$, bending the x component with x_{size} being the size of the section in x-direction. $\beta \neq 0$ determines the amount of curvature applied. If more than one zone is used, each zone has its own parameter β, describing its curvature. Additionally, a twist around the x axis allows us to describe asymmetric bending (see Fig. 5b) and an affine transformation T allows us to change the position and the orientation of the bended object. Let B be the vector, containing all βs, t and the components of the affine transformation.

Let $D \in \mathbb{R}^2 \to \mathbb{R}, (x, y) \mapsto depth$ be the depth image of the Kinect camera mapping each pixel to its depth value. Let $P \in \mathbb{R}^3 \to \mathbb{R}^2$ be the corresponding projection function from the world coordinate system into the camera image. Let $k \in \mathbb{R}^3$ be the Kinect depth camera center in the world coordinate system. The difference between the surface deformed by $\mathcal{N}$ or $\mathcal{B}$ and the observations of the depth camera can now be formulated as the difference between the depth in the depth camera measurement and the actual distance of the vertex to the depth camera center for each vertex v_i. Using the NURBS function, it can be written as:

$$\begin{aligned} \delta = \quad & D\left(P\left(\mathcal{N}_C(p_i) + o_i \cdot \bar{\mathcal{N}}_C(p_i)\right)\right) \\ & -\|\mathcal{N}_C(p_i) + o_i \cdot \bar{\mathcal{N}}_C(p_i) - k\|. \end{aligned} \tag{4}$$

The latter can be also expressed using the bending parameters B for the bending function $\mathcal{B}$:

$$\delta = D\left(P\left(\mathcal{B}_B(p_i)\right)\right) - \|\mathcal{B}_B(p_i) - k\|. \tag{5}$$

Hence, the search for the NURBS parameters C^* producing the smallest RMS in the observation can be denoted as

$$C^* = \arg\min_C \sqrt{\sum_i \delta^2} \tag{6}$$

or for the bending $\mathcal{B}$, the best parameters B^* can be denoted as

$$B^* = \arg\min_B \sqrt{\sum_i \delta^2}. \tag{7}$$

For a more detailed description of the NURBS deformation model, please refer to [12]. For more information on the bending model, please refer to [16].

The minimization task is solved using the CMA-ES [40] optimization method (Covariance Matrix Adaptation - Evolution Strategy), a particle filter like algorithm which is able to circumvent possible local minima in the consistency costs.

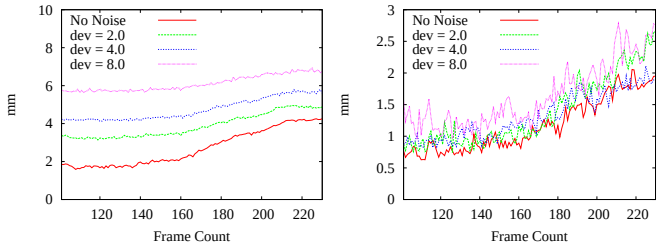


Fig. 6: Results from an artificial pick-up and bend sequence using NURBS tracking, simulated with various noise levels. Left: the RMS depth error between input data and synthesis. Right: The actual RMS difference of each object vertex between the simulation and the tracking results (Note the different scales of the right and the left plot).

Results: To validate the tracking method, tests on simulated and real data were performed. The absolute accuracy was tested by rendering a depth and color image sequence (40 cm distance between camera and object, 640×480 pixel resolution) from a deforming 3D model ($15\ \text{cm} \times 6\ \text{cm} \times 0.8\ \text{cm}$), simulating a pick-up and bend action (similar to the real object depicted in Fig. 7). This image sequence was used as input for the tracking algorithm using the NURBS deformation function. To test the robustness to noisy image data, different levels of noise were added to the depth image.

Fig. 6a depicts the RMS depth error for each frame of the tracking sequence when Gaussian noise is added with a deviation of 2, 4 and 6 mm The RMS depth error was calculated regarding all object pixel depth values of the synthesized object. The plot shows a clear dependency between the input noise level and the RMS depth error. Since ground truth data is available for the given input image, the tracking result was also compared to the object the image sequence was rendered from. Fig. 6b plots the RMS distance between the tracking result and the original object for the same sequence. The influence of noise in individual measurements to a global optimization goal calculated from many measurements can be small if the noise is not systematic or highly correlated; the more measurements available or the more restrictive the optimized model, the more robust the optimization goal becomes. It shows that the actual tracking error is below the RMS depth error and that the algorithm is far less sensitive to noise than the RMS depth error might suggest.

To test the algorithm on real input data, a deformation sequence similar to the synthetic test was recorded with a Kinect camera. Fig. 7 depicts 3D renderings of the input footage along with the NURBS deformation function approximating the surface very roughly. The resulting depth error is depicted in Fig. 8 (red line). Fig. 8 also displays the quantization step size of the Kinect camera at the average object distance (blue line) giving a hint on the depth sensor accuracy. The green line in Fig. 8 plots the RMS depth error when results are calculated in real-time with less iterations. A study on how to speed up the CMA-ES search for realtime usage of this algorithm can be found in [13].

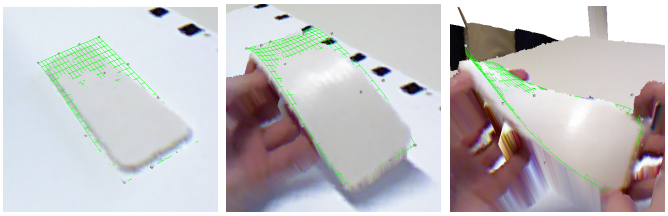


Fig. 7: The tracking sequence from real input data. The images show a colored 3D mesh, based directly on the Kinect input data along with the deformation function (green grid) roughly approximating the object surface.

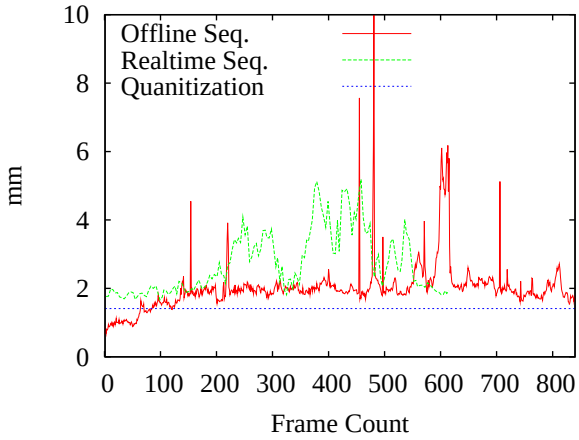


Fig. 8: Results from a real tracking sequence. The blue line depicts the quantization of the Kinect camera at the average object distance. The red line shows the RMS depth error when the object is tracked offline, the green line depicts the RMS depth error of the online tracking results.

The evaluation of the tracking algorithm using a bending-deformation function instead of the NURBS function can be found in section III-B.

Conclusion and relation to other modules: The described Kinect based vision system is able to track the deformation of flexible objects in real time with an excellent handling of sensor noise. The tracking directly yields the parameters of a deformation and is easily extended with additional constraints or physical models as e.g. discussed in section III-B to sense material properties.

B. Sensing of Material Properties

A linearly elastic, isotropic material may be characterized by two position-dependent stiffness parameters, e.g. Young's modulus $Y(r)$ and Poisson's ratio $\nu(r)$ [3]. When these parameters are known, along with the mass density $\rho(r)$, the geometry of the object and a relevant and consistent set of boundary conditions, it is possible to solve for the deformation by the means of analytical or numerical methods. To solve the problem of estimating the material properties, along with other important parameters, the sensing of material properties combines vision (section III-A) and modeling (section III-C).

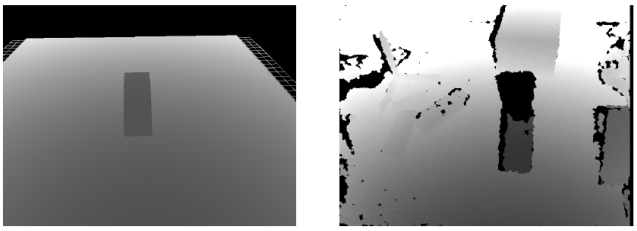


Fig. 9: Depth camera image of the object showing the synthesized version (left) and the real input data (right) as used in the estimation of material properties. Each image pixel represents a distance measurement, the brighter the pixel, the farther away the observed surface. Black pixel are labeled as undefined.

To evaluate physical correctness for a deformation, a residual function is formulated.

With a parameterizable deformation model at hand (as outlined in section III-A) and a residual function that provides a measure for physical plausibility for given deformation parameters B and material property Y, an overall measure can be defined that also takes the optical depth and color input into account. Similar to [12], an error function is defined which synthesizes a view of the deformed model and compares it to the input data (analysis by synthesis). The synthesis is done by rendering the object subject to the deformation $\mathcal{B}_B$ with projection parameters equivalent to the projection of the real camera (see Fig. 9).

The joint error value of the physical model and camera data is constructed by combining the color error e_c, the depth error e_d (see [12] for more details) as well as a physical error e_r. The physical error is given by the difference between the z-component of the deformation state $\mathcal{B}_B(s)$ (as introduced in section III-A) and the 1D deformation curve calculated by the physical model $\eta_Y(s)$ (see section III-C):

$$e_r = \frac{1}{|S|} \sum_{s \in S} (\eta_Y(s) - \mathcal{B}_B(s))^2, \tag{8}$$

where S is the set of all slices in the mesh (see Fig. 5). The error e_r is thus a measure on how well the current guess for a deformation state B and material property Y match the physical model.

A joint error function mapping B and Y to its overall plausibility can now be defined, by the combination of the error values e_r, e_c, and e_d:

$$f(B, Y) = \sqrt{\lambda_r e_r^2 + \lambda_c e_c^2 + \lambda_d e_d^2}\ . \tag{9}$$

As in [12], the error function (9) is minimized using CMA-ES. The resulting parameters provide the deformation parameters most suitable to the given input data.

Results: To evaluate the performance of our estimation we perform tests on synthetic and real data. For the synthetic data we render a triangle mesh using ground-truth data from a large-deformation finite-element simulation, allowing different Young's modulus to be simulated. For the real data, we estimate the Young's modulus for differently sized objects, for which we as a reference use an uniaxial tensile test establishing the relationship between the tension and the elongation in the test specimen.

Y ($\mathrm{N/mm^2}$)	offset error (mm)	Y error ($\mathrm{N/mm^2}$)
0.5	4.945	0.062
2.0	2.638	0.116
8.0	11.10	1.98

TABLE I: Estimation results for synthetic data.

The results for our tests on synthetic data can be seen in table I. In our experiments with the synthetic data we see a low RMS error of both Young's modulus Y and the offset distance for objects that have a high deflection. For objects that do not provide much deformation, such as the one with $Y = 8.0\ \mathrm{N/mm^2}$, our estimation does not converge to the correct results. This is reasonable as for higher values of Y there is little change in the object movement and as a result little change in the physical error e_r which in turn makes optimization difficult due to various error sources. As a consequence it can be expected that for stiffer objects, our estimation diverges from the true result.

Object	Y ($\mathrm{N/mm^2}$)	Y estimated ($\mathrm{N/mm^2}$)
7 mm	0.50	0.40
10 mm	0.45	0.45
13 mm	0.45	0.45

TABLE II: Estimation results for real data.

The results for the tests on real data is summarized in table II. From our results for the real data it can be seen that Youngs modulus have been acquired with an average error of $Y = 0.1\ \mathrm{N/mm^2}$ and below, compared to tensile tests.

While we have shown how to estimate Young's modulus and pose for a homogeneous object using visual data, there are issues that should be addressed for practical applications. First, objects encountered in the real world may be non-homogeneous e.g. a piece of tissue may have hard pieces of bone in it. While our system supports a position-dependent Young's modulus, the estimation will be non-unique as we derive an average mass density from the measured volume and weight of the object. Second, the simple Euler-Bernoulli beam equation used in the estimation algorithm does not incorporate the Poisson effect. Therefore for non-zero ν we are actually estimating an effective Young's modulus rather than the true material constant. Using a more accurate model, e.g. the non-linear beam model presented in section III-C, will obviously incorporate the effect. However, [31] demonstrated that the Poisson effect has very little influence on the observable result for volumetric measurements using visual data. Since we in our case are only observing and fitting our model to 2D deformation we will not be able to uniquely identify both Y and ν.

For a more detailed description of the estimation process and the experimental verification, please refer to [16].

Conclusion and relation to other modules: In this subsection, we described the sensing of material properties from the

visual data computed in section III-A. The estimated material properties are then used to find optimal action trajectories (as described in section III-D) based on the simulations as described in the next sub-section.

C. Modeling and Prediction

The full numerical solution of a continuous 3D object is in general a computationally hard problem [31]. Physically discretized models such as mass-spring models are often very fast and efficient, but it can be difficult to relate physical parameters to the stiffness of the springs [33], something which makes it harder to validate and switch to different models. In our work, we choose the class of realistic, mathematically discretized models of 2D deformation commonly known as beam models.

1) Linear Beam Models: To facilitate the different precision and runtime needs required by the individual stages of the system, we employ models of different degree of complexity. For instance, the stage performing the initial picking of the deformable object (Fig. 1b) requires only low-precision, however, there is a time constraint for the object to not drop off the conveyor. This stage calculates the path that the object will take, and as such requires several calls to the deformation model. For this, we use an analytical solution of the homogeneous, constant cross-section Euler-Bernoulli beam solved for fixed-free boundary conditions [3].

For medium-precision tasks, which are still required to be used in an online manner we use the static Euler-Bernoulli beam model [41] formulated for inhomogeneous materials with non-constant cross-sections:

$$\frac{\partial^2}{\partial x^2}\left(Y(x)J(x)\frac{\partial^2\eta(x)}{\partial x^2}\right) = q(x), \tag{10}$$

where $\eta(x)$ is the transverse deflection (see Fig. 10), $Y(x)$ is Young's modulus as previously described, $J(x)$ is the second moment of area and $q(x) = g_2\rho(x)A(x)$ where g_2 is the vertical component of the directional vector of gravity and $\rho(x)$ and $A(x)$ respectively is the mass density and area for the cross-section.

The non-constant material parameters, varying cross section and varying body load in the governing equation, require the usage of a numerical method to solve the differential equation. For that equation (10) is discretized by means of second-order accurate, centered finite differences and solved using an implicit scheme. The output of the numerical method is the deformation $\eta(x)$ for the neutral surface of each mesh slice. To reduce the artifacts resulting from the usage of a linear strain tensor with larger deformations in the free-hanging case, a simple length compensation approach [42] is used for post-processing. This also recovers the deformation in x namely $\xi(x)$, assuming no-stretch in the longitudinal direction.

2) Non-linear Beam Model: For larger deformations the linear models are not sufficient as they do not capture higher-order effects. In particular they make the simplifying assumption of a linear strain tensor. A non-linear beam model, based upon [4], is formulated by the following deformation function

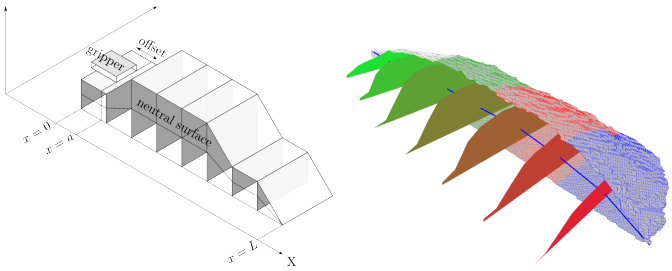


Fig. 10: Left: A gripper grasps a deformable object, with a gripper offset of a. Right: A triangle mesh as used in the system.

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \xi(x) - y\sin(a(x)) - x \\ \eta(x) + y\cos(a(x)) - y \end{bmatrix} \tag{11}$$

where $\xi(x)$ and $\eta(x)$ are deformations of the neutral surface in the x and y directions respectively and $a(x)$ is the angle of the neutral surface wrt. the undeformed configuration. With $\xi'(x) = \cos(a(x))$ and $\eta'(x) = \sin(a(x))$ we now write the total gravitational potential energy for a beam of length L as

$$E^g = \int_0^L \{g_1[b_0(x)(\xi(x) - x) - b_1(x)\eta'(x)] + g_2[b_0(x)\eta(x) + b_1(x)(\xi'(x) - 1)]\}\, dx \tag{12}$$

where $b_0(x)$, $b_1(x)$ are constants that depend on the integral of mass density over the cross section. The constants g_1 and g_2 are the components of the directional vector of gravity e.g. for a horizontal placement of the gripper, $\boldsymbol{g} = (0, -9.82)$.

We include geometric non-linearities by using the large-deformation strain tensor including higher-order terms [3]

$$u_{ik} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i} + \frac{\partial u_l}{\partial x_i}\frac{\partial u_l}{\partial x_k}\right) \tag{13}$$

where u_i are the components of the deformation function (11).

The elastic free energy for a unit volume of an isotropic material is generally defined [3] as

$$e^e = \frac{1}{2}\lambda u_{ii}^2 + \mu u_{ik}^2 \tag{14}$$

where $\lambda = \frac{Y\nu}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{Y}{2(1+\nu)}$ are the Lamé coefficients [3], and u_{ik} the strain tensor defined in (13).

With our deformation functions of (11) we have that the only non-zero element of the strain tensor is u_{11}. Assuming conditions of plane stress the total elastic potential energy in the beam becomes

$$E^e = \int_0^L \left(4c_2(x)a'(x)^2 + 4c_3(x)a'(x)^3 + c_4(x)a'(x)^4\right) dx \tag{15}$$

where $c_2(x)$, $c_3(x)$, $c_4(x)$ are constants which depend on the integral of the elastic moduli over the cross section.

By direct computation we can now eliminate the terms of $\xi(x)$ and $\eta(x)$ and formulate the total potential energy $E^t = E^g + E^e$ as a function of the angle $a(x)$ only:

$$
\begin{aligned}
E^t = \int_0^L & \{(g_1 B_0(x) + g_2 b_1(x))\cos(a(x)) \\
& + (g_2 B_0(x) - g_1 b_1(x))\sin(a(x)) \\
& + 4c_2(x)a'(x)^2 + 4c_3(x)a'(x)^3 \\
& + c_4(x)a'(x)^4 - g_1 b_0(x)x - g_2 b_1(x)\} \, dx \\
\equiv & \int_0^L f(x)dx
\end{aligned} \tag{16}
$$

where $B_0(x) = B_0^*(L) - B_0^*(x)$ with $B_0^*(x) = \int_0^x b_0(s)ds$.

By the principle of stationary action, we determine the deformed beam shape by minimizing the total potential energy of (16) and subsequently integrating the solution vector a_i for $i \in [0; M]$ with $a_i \approx a(ih)$ where h is the step size. To approximate the derivatives we employ second-order accurate, centered FD expressions for the internal points and forward/backward differences at the endpoints. Using the trapezoid rule we approximate (16) by

$$
E^t = \frac{h}{2}\left[f(0) + f(Mh) + 2\sum_{i=1}^{M-1} f(ih)\right] \tag{17}
$$

where f is the integrand from (16) and M is the number of points in the discretization.

The boundary conditions are satisfied by formulating suitable constraints for the optimization. E.g. to simulate contact with the table in the Laying-Down task, a constraint is added to the optimization which ensures that no points on the beam are allowed to penetrate[1].

3) Results: For our experiments presented in this paper we have employed the robot framework RobWork [43] and the IPOPT optimization package [44] using the MUMPS direct solver [45], [46]. For the high-precision, offline learning of the Laying-Down task, we build a database of known objects. We perform a sweep over the height and the angle of the gripper and for each configuration we calculate the deformed shape of the object and the total elastic energy. Should a new object be introduced into the system, new tables are generated and the learning is trained again. The results from this task are presented in section IV-C.

To estimate the actual elastic energy in objects during the Laying-Down task, we capture a 3D point cloud using the Kinect camera during path execution. Outliers are subsequently filtered and the data is fitted by linear least squares to third-degree polynomials. These polynomials are then inserted into (15) and the energy is evaluated by numerical integration. A comparison between the estimated energies and those obtained from the simulation, can be found in section IV-C.

In order to evaluate the accuracy of the simulation, we have compared the output of the simulation with Kinect surface data. We used the estimated value for Young's modulus of $Y = 0.40\,\mathrm{N/mm^2}$ for the 7 mm thin object. Fig. 11 shows the comparison for 4 different poses in a Laying-Down action.

In general we had good correspondence between the simulation and observed surface data for poses with the object being either free-hanging or relatively horizontal against the table. However, in some situations the object is pushed in its longitudinal direction into the table (Fig. 11, pose 3) causing a buckling behaviour not captured by the simulation. This is caused by a high coefficient of friction between the silicone rubber and the wooden table, making objects stick. The assumption of the end being free to move is therefore invalid causing the simulation and the real world results to diverge. While possible to include friction effects in the simulation, experience has shown us that the friction between flexible objects and the table is highly variable and as such difficult to parameterize.

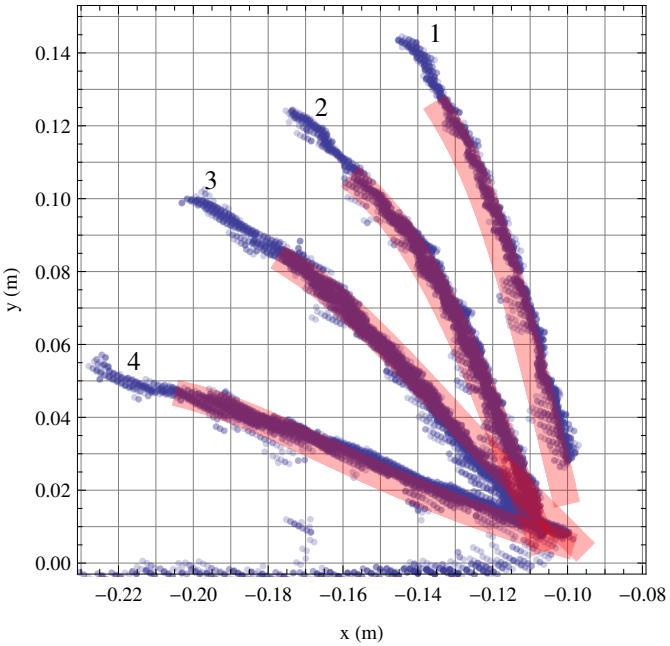


Fig. 11: Plot of raw Kinect 3D vertex data versus simulated deformation for 4 different poses in a Laying-Down action. The blue points show the vertex surface data and the red area is the corresponding simulated deformation. The cluster of points at $y = 0$ are boundary artifacts in the Kinect data.

Conclusion and relation to other modules: We have demonstrated a good fit between simulated and real deformations, which gives indications that actions performing well in simulation will also perform well in reality. Hence simulation can be used to find promising actions. However, with limitations of the deformation prediction in capturing effects such as friction and also inaccuracies of the vision and material sensing system (as outlined in section III-A and III-B), there is a need to perform learning also in the real world. The computation of promising actions in simulation as well as the further fine-tuning of these actions in a real world scenario are described in the next sub–section.

D. Learning in Simulation and Real World Execution

When addressing a learning problem, the exploration of the search space is an essential aspect, as a high degree of exploration increases the likelihood for finding optimal solutions for the problem. However, doing experiments in

[1]For more details about the non-linear beam model, please refer to [42].

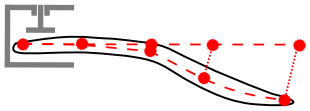


Fig. 12: Illustration of the feature vector based on a 1×5-dimensional grid.

the real robot setup is time-consuming and thereby costly. Therefore in our approach, the learning process is bootstrapped with simulated experiments based on the modeling described in section III-C. Subsequently real experiments are used to adjust the learned model.

In section III-D1, the parametrization of objects is defined (common for both actions, Peg-in-Hole and Laying-Down). The methodology for the learning of these two actions is described in section III-D2 and III-D3.

1) Object Parametrization: The parameterized description of objects is defined to allow for the generalisation of actions across similar objects. As the deformation characteristics of an object is essential for the selection of an optimal action, the difference between the shape of the undeformed object S_o and the shape S_d of the object when it is deformed based on the impact of gravity while being fixated at a horizontal position is estimated:

$$\hat{S}(u,v) = S_o(u,v) - S_d(u,v), \tag{18}$$

where (u,v) are coordinates on a grid of size $I \times J$. The deformations are obtained at a set of discrete locations and form a feature vector $\mathbf{f}$. As only a two-dimensional deformation is modeled (see section III-C), a 1×5-dimensional vector is used[2] (illustrated in Fig. 12):

$$\mathbf{f} = \frac{1}{L}\left[\|\hat{S}(g_0)\|, ..., \|\hat{S}(g_i)\|, ..., \|\hat{S}(g_I)\|\right] \in \mathbb{R}^I, \tag{19}$$

where L is the length of the object. However, the feature vector can easily be extended to cover more complex deformations, e.g. by using a 3×5-dimensional grid to cover the twist of the object.

The aim with the formulation of $\mathbf{f}$ is to facilitate the comparison of object characteristics, such that similar actions can be applied to similarly behaving objects. As sharing properties such as geometry, appearance or mass do not guarantee similar characteristics, such properties are not used.

2) Peg-in-Hole Actions: The actual Peg-in-Hole action is defined as a trajectory $P(t) \in \mathbb{R}^2 \times SO(2)$ (see figure Fig. 13b). The target configuration P_1 is known as the gripper will be directly in front of the hole with a horizontal orientation. The starting configuration P_0 is based on the deflection modeling, chosen such that the tip of the object is in front of the hole and the surface tangent at the tip is close to horizontal (see Fig. 13a). The trajectory (illustrated on Fig. 13b) defining the interpolation between the starting and target configuration is defined using a rational Bézier-curve [47] based on three points, namely the start and end configurations and an additional control point, $\boldsymbol{\omega}_P = [\omega_P^x\, \omega_P^z\, \omega_P^\alpha]$ (the superscript indicates the affected dimension, illustrated in Fig. 13a):

[2]We assume the stretch of the object in the longitudinal direction to be negligible, thus allowing the deformation at a point to be described by a single scalar value.

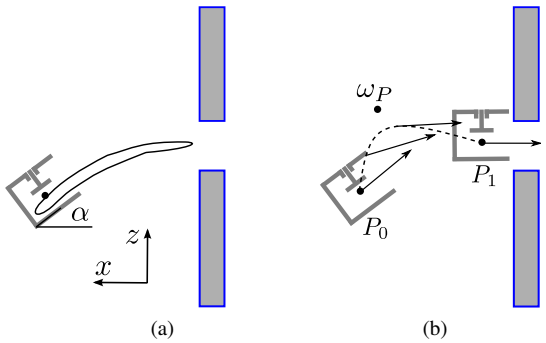


Fig. 13: Illustration of (a) starting configuration P_0 for the Peg-in-Hole action. (b) shows a projection of the trajectory in $\mathbb{R}^2 \times SO(2)$ based on P_0, the target configuration P_1 and the control point $\boldsymbol{\omega}_P$. The arrows indicate the orientation of the gripper along the path.

$$P(t) = P_0 + B(t) \circ (P_1 - P_0) \text{ for } t \in [0;1], \tag{20}$$

with

$$B(t) = \frac{\sum_{i=0}^{n} b_{i,n}(t)\alpha_i \boldsymbol{\omega}_i}{\sum_{i=0}^{n} b_{i,n}(t)\alpha_i}, \tag{21}$$

where $b_{i,n}(t)$ is the Bernstein polynomial with $n = 2$, $\boldsymbol{\omega}_i \in \{\mathbf{0}, \boldsymbol{\omega}_P, \mathbf{1}\}$ refers to the i'th control point for the curve and $\circ$ denotes the entrywise (Hadamard) product.

The weights $\alpha_i \in \{1, 2, 1\}$ are fixed, which ensures that the second control point, which is subject to learning, has an increased impact.

The modeling of the potentially successful Peg-in-Hole actions is inspired by Kernel Density Estimation (KDE) [6]. Every time a control point that leads to a successful action has been obtained, it is added to a density d. However, contrary to the situation in [5], where grasp affordances are learned for a specific object, we cannot assume the objects to be identical (see section III-D1). Therefore, a kernel, $\mathbf{K}_{\mu,\sigma}(\boldsymbol{\omega}_P, \mathbf{f})$, which is a compound of two kernels is used: one reflecting the Peg-in-Hole action as such, the other reflecting the object parametrization specified in (19):

$$\mathbf{K}_{\mu,\sigma}(\boldsymbol{\omega}_P, \mathbf{f}) = \mathbf{N}^{PiH}_{\mu_p,\sigma_p}(\boldsymbol{\omega}_P)\, \mathbf{N}^{Object}_{\mu_f,\sigma_f}(\mathbf{f}) \tag{22}$$

$$\text{with } \mu = \{\mu_p, \mu_f\} \text{ and } \sigma = \{\sigma_p, \sigma_f\} \tag{23}$$

where $\mathbf{N}^{PiH}$ and $\mathbf{N}^{Object}$ are isotropic multivariate Gaussian kernels located at the mean positions μ_p resp. μ_f and with a bandwidth of σ_p resp. σ_f. Note that μ_p corresponds to the 3-dimensional control point of an evaluated action and μ_f corresponds to the currently 5-dimensional feature-vector of the object that was involved in that action.

The density d_{PiH} is given by the weighted sum of the kernels given m experiments:

$$d_{PiH}(\boldsymbol{\omega}_P, \mathbf{f}) = \eta \sum_{i=1}^{m} \mathrm{w}_i \mathbf{K}_{\mu_i,\sigma}(\boldsymbol{\omega}_P, \mathbf{f}), \tag{24}$$

where w_i is a weight associated with each sample and η is a constant ensuring that the density integrates to 1. In order to condense the notation, $\mathbf{K}_i$ replaces $\mathbf{K}_{\mu_i,\sigma}$ in the following.

The learning of the density has been initiated in simulation (see section III-C for further details on the simulation) where the clearance during the operation can be easily determined. The thickness t_i of the object and the minimal clearance c_i which has been observed during the operation are used to define the individual weights:

$$\mathrm{w}_i = \begin{cases} \frac{t_i + c_i}{\sum_{j=1}^{m} \mathbf{K}_j(\boldsymbol{\omega}_{Pi}, \mathbf{f}_i)} & \text{if } c_i > 0 \\ 0 & \text{else} \end{cases} \tag{25}$$

where the denominator counteracts the otherwise accumulating effect of nearby samples.

As a consequence, when an action for a new object with feature vector $\mathbf{f}_{new}$ and thickness t_{new} is considered, the expected clearance $\hat{c}$ is given by:

$$\hat{c} = \frac{1}{\eta} d_{PiH}(\boldsymbol{\omega}_P, \mathbf{f}_{new}) - t_{new}. \tag{26}$$

However, in the real setup the clearance cannot be measured — only the fact whether an action failed or not is accessible. Therefore, two different use cases for the density based on simulated experiments exist: 1) for a given object, the most promising action can be estimated and executed and 2) specific regions can be explored further with real experiments and lead to a correction of the density.

The most promising action, i.e., the action with the largest expected clearance, can be found by searching for a global maximum of the density given the feature vector for the object for which the action is to be selected (see [17], [48] for details). Note that, since $\mathbf{f}$ is given by the object at hand, finding the maximum of the density is only a 3D search problem.

When a specific control point is evaluated on the real system, the density can be used to estimate whether the clearance is expected to be above zero or not based on equation (26). If the action leads to a collision although a nonzero clearance was predicted or vice versa, it is an indication for the density to be incorrect at this point. Such errors are reduced by adjusting the individual weights accordingly:

$$\Delta \mathrm{w}_i = -\tau \mathbf{K}_i(\boldsymbol{\omega}_P, \mathbf{f})\, k \tag{27}$$

$$\text{with } k = \begin{cases} |\hat{c}| & \text{if } c > 0 \text{ and } \hat{c} < 0 \\ |\hat{c}| & \text{if } c < 0 \text{ and } \hat{c} > 0 \\ 0 & \text{else} \end{cases} \tag{28}$$

where τ is a learning rate and the evaluated action consisting of the control point $\boldsymbol{\omega}_P$, feature vector $\mathbf{f}$ and the true clearance c is used to adjust the density if the expected clearance $\hat{c}$ is wrong. Such an update rule can be applied in a situation where the system is running on a long term basis and a scheme controlling the trade off between a high performance (always selecting the control point that maximizes the density) and exploration, enabling the system to improve. However, such longterm aspects are outside our scope and essentially not tractable with the setup at hand as it was not optimized with respect to speed and the evaluation of an action currently takes about one minute. Therefore, the setup was tested as a whole with a single object, allowing for exploration for this specific object. Subsequently a new optimal control point was estimated for this object, see section IV-B.

Results: The simulations have been performed for different objects with 8000 samples, spanning an equidistant grid with a resolution of 0.1 in each dimension,for each object. A visualization of the density based on the actions on a single object (with a thickness of 7 mm) is shown on Fig. 14. It becomes evident that the region of successful actions is roughly convex, which justifies the search for a maximum. Regions where no successful actions have been recorded are clamped to zero. Note that, locally, the value of the summed clearance and thickness might be lower than the real thickness of the objects — this is caused by the smoothing effect introduced by KDE. The bandwidth $\boldsymbol{\sigma}_P = 0.004\, I_3$, with I_3 being a 3×3 identity matrix, was kept isotropic and chosen manually to ensure a sufficiently smoothed density. As only a single object is considered in the test, the choice of $\boldsymbol{\sigma}_o$ is not relevant.

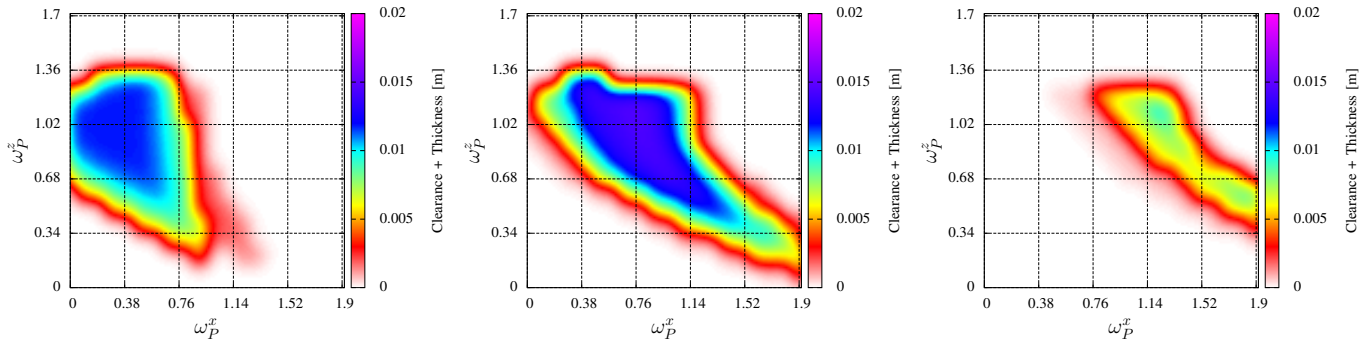
Further, real-world Peg-in-Hole actions are performed with one object matching a simulated one. In total 80 different actions, exploring the region that also contains the maximum (illustrated in Fig. 15a), have been performed and each action has been repeated 10 times. The control points were selected using a coarser 2-dimensional grid with an resolution of 0.2, keeping ω_P^α fixed. An exploration of all 3 dimensions was not tractable.

As the outcomes in simulation and real-word experiments do not correspond directly — in simulation the clearance is determined, in real-world experiments only successes or failures are distinguished — they do not facilitate a direct comparision. However, obviously all actions with non-zero clearance are expected to succeed. Therefore, the distributions of simulated actions with non-zero clearance is expected to be similar to the distribution of succeeding real-world actions – which indeed is the case (see Fig. 15). The distribution of the N real-world actions is modeled using (24) with $\boldsymbol{\sigma}_P = 0.012\, I_3$ just as the simulated actions. The weights w_i are defined slightly different:

$$\mathrm{w}_i = \frac{o_i}{\sum_{j=1}^{N} \mathbf{K}_j\,(\boldsymbol{\omega}_{Pi}, \mathbf{f}_i)} \text{ for } o_i \in \{0,1\}, i \in \{1,\ldots,N\}, \tag{29}$$

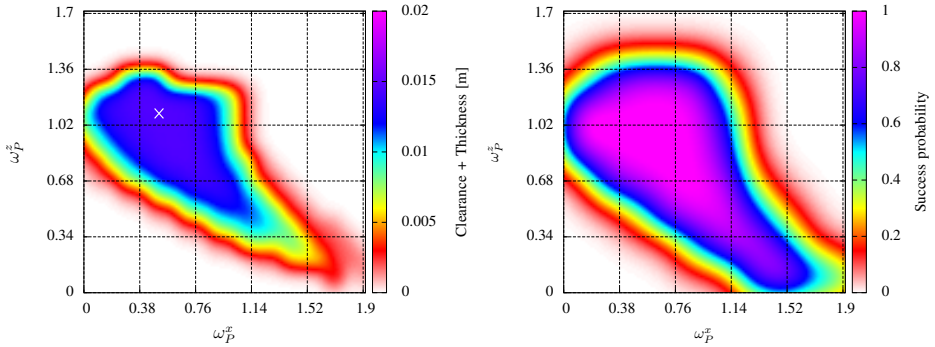
where $o_i = 1$ indicates a success and $o_i = 0$ indicates a failure.

3) Laying-Down Actions: Similarly to the Peg-in-Hole action, the Laying-Down action is defined as a trajectory in $\mathbb{R}^2 \times SO(2)$. The aim of the operation is to place the object at a pre-defined target while minimizing the internal forces of the object (caused by deformations) as too high forces in general might damage the object. In the following we will derive an objective function which will be optimized to find an optimal action by making use of two aspects: First, the maximal energy applied to the object during the manipulation process and second, the time required for the manipulation in terms of the length of the travelled trajectory. In addition, the stability of the simulator is incorporated in order to identify if



(a) $\omega_P^\alpha = 0$ (b) $\omega_P^\alpha = 0.32$ (c) $\omega_P^\alpha = 0.63$

Fig. 14: Visualization of a density based on simulated experiments for one object.



(a) Simulated Peg-in-Hole actions, $\omega_P^\alpha = 0.20$ (b) Real-world Peg-in-Hole actions, $\omega_P^\alpha = 0.20$

Fig. 15: A density based on simulated experiments for one object, (a) shows a slice of the density with the global maximum highlighted by a white cross. (b) illustrates the average outcomes (success or failure) of corresponding real-world experiments.

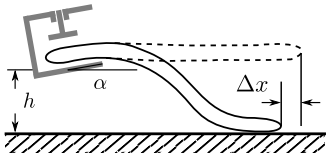


Fig. 16: Illustration of the parameters h and α defining the Laying-Down operation. The displacement Δx of the tip, caused by the deformation, is estimated based on the modeling.

specific parameter regions cannot be simulated with sufficient accuracy.

The beginning of the trajectory is defined such that the tip of the object touches the target, subsequently the gripper guides the object to the target location, ideally without causing the tip of the object to move. The trajectory is therefore characterised by the height h and tilt α of the gripper relative to the table as well as a horizontal displacement Δx (see Fig. 16). Given h and α the deformations of the object and the elastic energy E^e in the object can be predicted as outlined in section III-C3. To ease the notation, we consider the following function to be available:

$$P_x(h, \alpha) \rightarrow \Delta x. \tag{30}$$

Pretending that the table is frictionless, the predicted displacement of the tip, Δx, is transferred to the gripper in order to prevent a displacement. As the true friction coefficient between the object and the target surface is unknown, minimizing the movement of the object tip relative to the target surface also minimizes the impact of the unknown friction properties. Therefore the learning problem is only a 2-D problem, addressing the height h of the gripper and it's orientation α with respect to gravity. which is used to define the trajectory in terms of a parametric function:

$$P(t) = \{x(t), h(t), \alpha(t)\} \tag{31}$$

$$= \{L - P_x\left(h(t), \alpha(t)\right), h(t), \alpha(t)\} \text{ for } t \in [0; 1] \tag{32}$$

where L is the length of the object,

$$\begin{bmatrix} h(t) \\ \alpha(t) \end{bmatrix} = \begin{bmatrix} h_1 \\ 0 \end{bmatrix} + B(t) \circ \begin{bmatrix} h_1 - h_0 \\ 1 \end{bmatrix} \tag{33}$$

and $B(t)$ is given by (21). The height of the gripper at the end and at the beginning of the operation is defined by h_1 and h_0 respectively. The three control points used by $B(t)$ are given by:

$$\boldsymbol{\omega}_i \in \left\{ \begin{bmatrix} 0 \\ \alpha_0 \end{bmatrix}, \boldsymbol{\omega}_L, \begin{bmatrix} 1 \\ \alpha_1 \end{bmatrix} \right\} \tag{34}$$

where $\boldsymbol{\omega}_L$ is optimized with respect to a combined criteria of minimizing both the maximal energy E^e during the action and the length Φ of the trajectory, in terms of joint angles, travelled by the robot:

$$g(E^e, \Phi, c) = \frac{1}{E^e} \frac{S(c)}{\Phi} \tag{35}$$

where c denotes the consistency of the simulated action indicating if simulations did not converge and $S(c)$ denotes a sigmoid function defined below. Note that the simulator is used for every step in the discretized trajectory. If the simulation fails or doesn't converge at a single step, an interpolation between the two adjecent steps can be done, however, if multiple steps fail, an interpolation might be misleading. Therefore, the sigmoid function $S(c) = \frac{1}{1+\exp(ac-b)}$ where a, b are constants is introduced to suppress regions in the search space where the simulation of an action failed. Conditions, such as the object being forced below the table, that challenge the simulation are of little practical relevance. Example data for the $7\ \mathrm{mm}$ thin object is provided in Fig. 17.

Similarly as for Peg-in-Hole actions, KDE is utilized to derive a continuous estimate of the distribution d_{LD} of Laying-Down actions based on the n simulations:

$$d_{LD}(\boldsymbol{\omega}_L, \mathbf{f}) = \eta \sum_{i=1}^{n} \mathrm{w}_i \mathbf{K}_i(\boldsymbol{\omega}_L, \mathbf{f}), \tag{36}$$

where the weights are defined as:

$$\mathrm{w}_i = \frac{g(E_i^e, \Phi_i, c_i)}{\sum_{j=1}^{n} \mathbf{K}_j(\boldsymbol{\omega}_{Li}, \mathbf{f}_i)} \text{ and } E_i^e = \max_t \left(P_e\left(h_t, \alpha_t\right)\right). \tag{37}$$

Results: When the Laying-Down operation is performed, the deformed object is tracked (as outlined in section III-A) and the elastic energy, caused by the deformation, is estimated based on the object shape (see section III-C). For three different control points, indicated in Fig. 17, approximately 50 actions have been performed on the real setup. The control points were chosen such that three different behaviours were observable: First a situation where the tip of the object is lifted from the table (outcomes are illustrated in Fig. 18b). Second, one with the object being pressed against the table (see Fig. 18c) and finally the third being close-to optimal (see Fig. 18a). The rightmost column contains the numbers of all actions with energies that exceed the limits of the axis – these are considered to be outliers.

It is noticeable that the usage of the control point that is optimal according to the simulations does not lead to the lowest energies in general. Instead, the control point that causes the object to lift from the table leads to lower energies with less variance. Those actions, where the object was pushed against the table (Fig. 18c), lead to highest energies with a large variance.

This variance is largely caused by the fact that the object in situations with contact to the table is sensitive to even tiny variations, e.g. caused by uncertainty in the grasp, letting the tip of the object either stick to the table, causing the object to buckle (also observed in Fig. 11), or just slide. The key results for the experiments are summarized in table III. When several Laying-Down actions with the same control point were evaluated, the estimated energies vary significantly. Therefore, the median of the energies $\tilde{E}^e$ are used. As the trajectory of the robot should be identical for several repetitions of the same actions, the average of the trajectory length $\overline{\Phi}$ was used.

$\boldsymbol{\omega}_L$	$\tilde{E}^e\,[10^{-6}\mathrm{J}]$	$\overline{\Phi}\,[\mathrm{rad}]$
a: ($0.24, -60$)	12.941	1.56
b: (-0.15, 20)	6.854	1.80
c: ($0.80, -53$)	18.734	1.59

TABLE III: Results of the Laying-Down actions using three different control points, illustrated in Fig. 17, evaluated on the real setup.

Conclusion and relation to other modules: Optimal trajectories for Peg-in-Hole and Laying-Down actions were first computed in simulation based one the modeling described in section III-C, the material parameters estimated in section III-B and an approximation of an appropriate objective function by a KDE approach.

After that, Peg-in-Hole actions were performed under ideal conditions – the result indicate a good match between the simulation and the real-world action. Further, three different Laying-Down actions were performed in order to support a qualitative comparison of simulated and real actions. The large variance of the results indicate that not all aspects are modeled by the simulation.

While in this section we have tested the sub–modules and the execution of Peg-in-Hole and Laying-Down actions under ideal conditions (in particular a controlled object grasping), we will in the following section evaluate the overall system and utilize the gathered real-world data perform a fine-tuning of both actions.

IV. Results of the Overall System

In this section, we will first describe the robot platform (section IV-A) and how the overall system performs when doing Peg-in-Hole (section IV-B) and Laying-Down actions (section IV-C).

A. Demonstrator

To facilitate realistic and reproducible tests with flexible objects, casted silicone models have been used. The material is a Dow Corning Silastic 3481 silicone rubber molded using Silastic 81-R hardener. The mass density for this is given to $\rho = 1.200 \cdot 10^{-6}\ \mathrm{kg/mm^3}$. Reference values for Young's modulus have been estimated by tensile tests using an industrial robot and a PASCO PS-2189 force sensor. The material and model shapes have been choosen in order be representative of cut meat from food production.

The robot used to handle the objects is a Universal Robots arm, UR5 — a lightweight industrial 6-DOF robot arm. Further, a small standard conveyor belt is used to transport the objects from an entry point, where the user inputs them, to the camera, where a 3D model is obtained and finally to the interaction stage, where the robot picks the object and

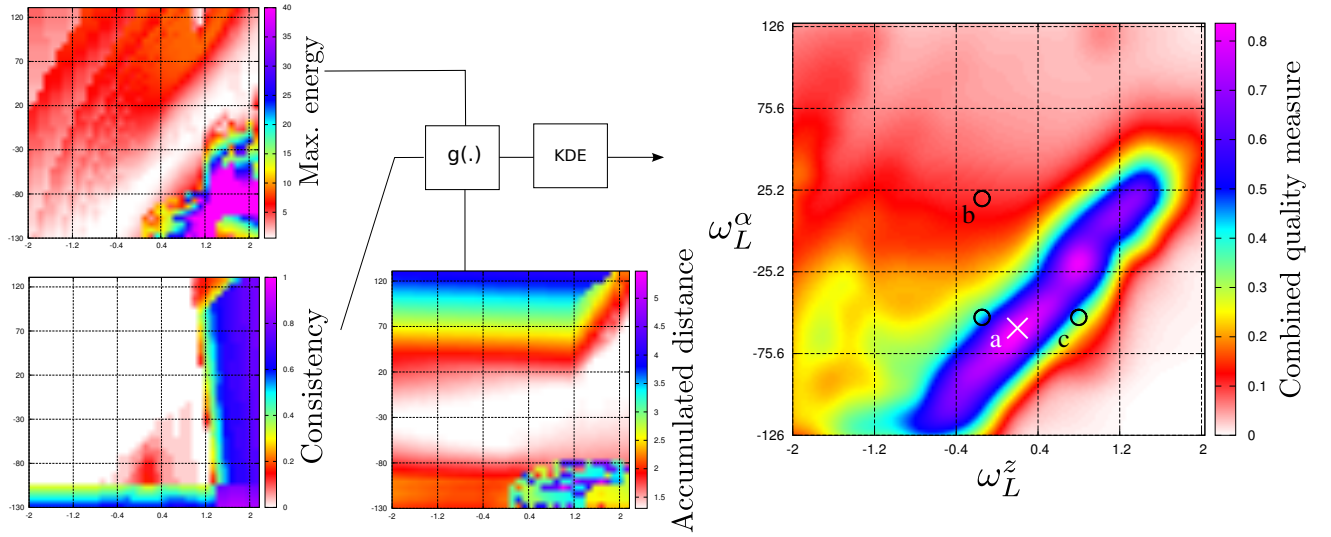
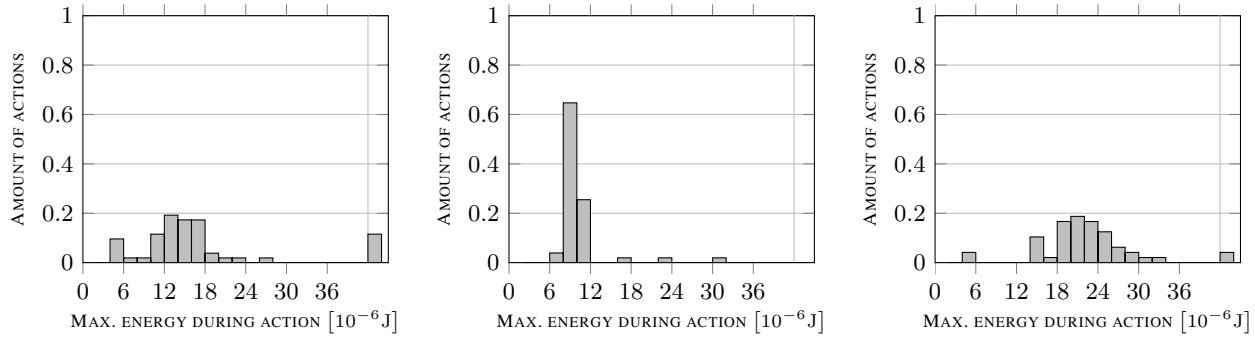


Fig. 17: The left plots illustrate different properties for simulated Laying-Down actions for the 7 mm thin object with respect to the 2D control point. Top left: maximal energy E^e during the action, bottom left: inconsistency of the simulation c and center: the accumulated distance in terms of joint angles Φ that the robot arm traveled. Right: combined quality measure, based on (35), where the maxima, corresponding to the optimal action, is highlighted by a white $\times$. The two black circles indicate control points that have been evaluated in addition to the optimal on the real setup. The outcomes are illustrated on Fig. 18, the labels indicates which histogram corresponds to the individual control points.



(a) $\boldsymbol{\omega}_L = (0.24, -60)$ (b) $\boldsymbol{\omega}_L = (-0.15, 20)$ (c) $\boldsymbol{\omega}_L = (0.80, -53)$

Fig. 18: For three different control points: The distribution of Laying-Down actions with respect to the maximal energy detected during the execution of the individual action. The index of the subfigures matches the overlayed letters in Fig. 17.

processes it. The gripper is a special 1-DoF design using a SMAC-actuator [49].

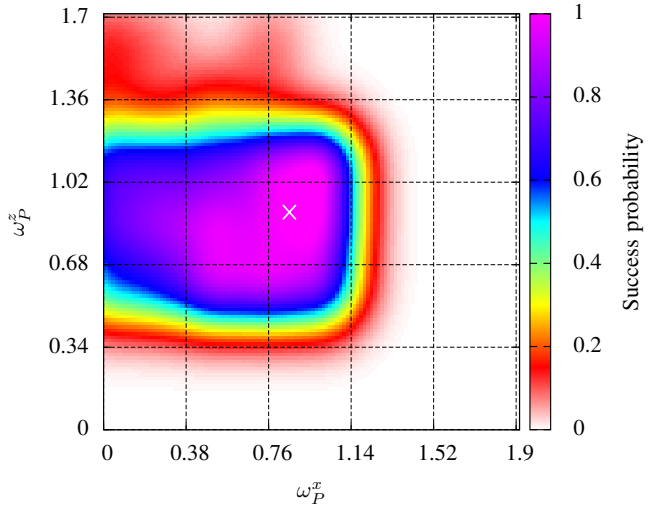
The mechanical part of the demonstrator for bundling all the different components (see Fig. 1) is made in lightweight materials such as framework in aluminum profiles, which allows fast adaptations and even fast dismantling and assembly moving the demonstrator to new facilities.

B. Results on Peg-in-Hole operations

In section III-D2 a good correspondence between simulated and real world Peg-in-Hole actions has been shown for the situation where the grasp applied to the object was ensured to match the simulated grasp. However, the grasp is subject to various error sources, such as the calibration of the set-up, the synchronization of the modules and in general error introduced by the vision system affecting the generated model. All errors will affect any successive steps and can hardly be included in the simulation. Therefore, the performance of the system is expected to be lower when the object is not grasped in a controlled, ideal fashion.

Using the 7 mm thin object, grasped when it reached the end of the conveyor belt, approx. 100 actions using the control point $\boldsymbol{\omega}_P^{max}$ that is ideal according to the simulated actions have been performed. Furthermore, approx. 250 actions, using control points on a grid in the vicinity of the previous control point, have been evaluated.

Subsequently all actions evaluated in real world experiments have been utilized to determine a new optimal action with control point $\hat{\boldsymbol{\omega}}_P^{max}$. This has been tested approx. 100 times — table IV shows an summary of all outcomes. Possible



(a) $\omega_P^{\alpha} = 0.202$

Fig. 19: The density based on real-world Peg-in-Hole experiments using a 5×5 grid with the maxima highlighted.

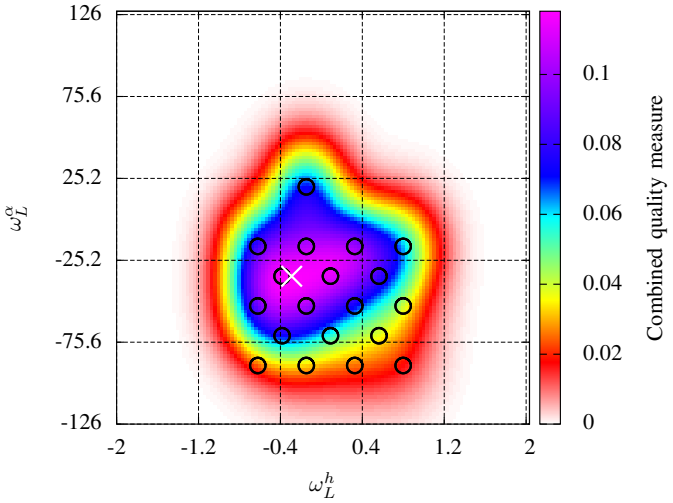


Fig. 20: Density based on real-world Laying-Down experiments using 19 different control points, indicated by black circles. The new maxima, corresponding to the optimal action, is highlighted by a white $\times$.

outcomes are success or failure, where failures cover *collision* or the path being *unreachable* for the robot. The latter is typically caused by the path planner rejecting configurations where collisions are expected. As an action that would lead to a physical collision either can lead to true collision or to a rejection by the path planning algorithm depending on e.g. tiny inaccuracies in the calibration, the two outcomes have been concatenated. In addition, the object might be *sliding* into the hole, which can be considered a failure as the object touched the brim of the whole, or a success as the object still has been inserted. In both cases the second learning phases, utilizing the real-world experiments, increased the performance, such that the optimal action succeeds in 92.9% resp. 99.1% of the attempts. Especially when *sliding* is considered to be a failure, the improvement compared to the results based on simulation only is significant.

control point	*success*		*collision* or *unreachable*	success rate (excl. *sliding*)
	no contact	*sliding*		
${\boldsymbol{\omega}_P}^{max}$	84	16	7	93.5% (78.5%)
${\boldsymbol{\omega}_P}^{avg}$	159	41	131	60.5% (52.7%)[3]
${\hat{\boldsymbol{\omega}}_P}^{max}$	104	7	1	99.1% (92.9%)

TABLE IV: Summary of the outcomes of Peg-in-Hole operations on the real setup using the 7 mm thin object.

C. Results on Laying-Down Operations

In section III-D3 and III-C3, it has been shown qualitatively that simulated and real Laying-Down actions correlate well. In the following, it will be investigated if the optimal action, determined using simulation, corresponds to the one that is optimal in real world experiments.

Using the 7 mm thin object, grasped when it reached the end of the conveyor belt, approx. 50 actions using each control point in the vicinity of the one that is optimal according to the simulations have been performed. During the execution of an action the actual shape of the object was tracked and used to estimate the energy as described in section III-C3. Furthermore, the trajectory of the robot arm was recorded. Thereby each evaluated action leads to a triplet consisting of a control point, the largest energy estimated during the action and the accumulated joint distance traveled by the robot arm.

As the estimated deformation energy relies on the mesh-models of the object in a given situation, outliers are expected to occur which might lead to wrong estimates of the energy. To reduce the impact of outliers, the median of all outcomes is estimated for each control point and used as the representative outcome. Based on these, the distribution of the quality with respect to the control point is estimated and a new maximum is determined (see Fig. 20).

The results were used to derive an updated estimate of the optimal control point which subsequently were evaluated — the distribution of the outcomes is illustrated in Fig. 21. Compared with the results for the previous estimate of the optimal control point, see Fig. 18a, in general lower energies were observed. However, when compared with the control point that leads to little contact between object and table, see Fig. 18b, often lower energies have been observed, but the median energies are comparable as the variance of the results for the optimal control point is significantly higher.

The control point that is estimated to be the optimal one leads to actions where the tip of the object is touching the table, such that the object is supported by both the table and the gripper. Apparently, the situations where the object touches the table are sensitive to even small deviations between the simulated model and the real world. Although the results indicate an improvement when the real-world data is utilized, they also indicate that additional sensors, for instance a force-torque sensor detecting when the contact between the target surface and the object occurs, probably would be beneficial.

[3]weighted by the distribution of samples

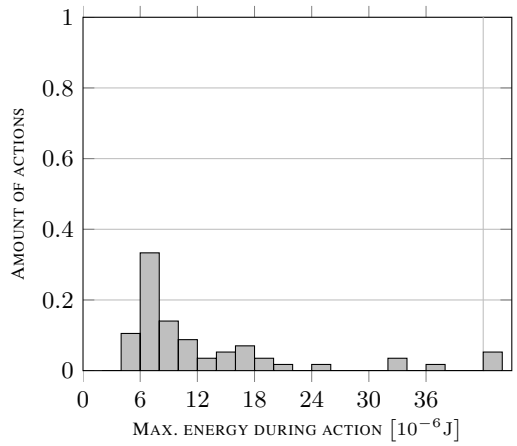


Fig. 21: Distribution of the maximal deformation energy observed during real-world Laying-Down actions when the optimal control point is used.

V. Conclusion

The overall task of the outlined system was to address the handling of flexible objects in robot workcells. The different submodules required for such a system were described, implemented and evaluated both individually and as a whole.

The proposed system achieves real-time sensing of flexible objects by direct, model-based tracking using analysis by synthesis of depth and colour images. Combining model-based tracking and deformation modeling, the elastic moduli and pose of grasped objects is determined. Having estimated the model parameters of grasped objects, the deformation is determined based on its current configuration using two different methods providing trade-offs between accuracy and computational effort. Learning has been employed to identify optimal manipulation actions in a simulated environment. Subsequently, the differences between the simulated and the real system are compensated by a secondary learning phase.

The proposed system combines the four modules and demonstrates their application to real-world scenarios. We have demonstrated our approach on conveyor belt grasping with subsequent Peg-in-Hole and Laying-Down manipulation actions. By that, we have provided a system in which vision, modeling, simulation and learning are combined to arrive at a flexible and adaptable system. We showed that the system is able to deal with the requirements of manipulation in not fully controllable contexts. This is in particular required for the manipulation of flexible objects.

References

[1] Scape Technologies - Standardized Bin-Picking Systems. [Online]. Available: http://www.scapetechnologies.com

[2] D. Kraft, L.-P. Ellekilde, and J. A. Jørgensen, "Automatic grasp generation and improvement for industrial bin-picking," in Gearing up and accelerating cross-fertilization between academic and industrial robotics research in Europe:, ser. Springer Tracts in Advanced Robotics, F. Röhrbein, G. Veiga, and C. Natale, Eds. Springer International Publishing, 2014, vol. 94, pp. 155–176.

[3] L. D. Landau, L. P. Pitaevskii, E. M. Lifshitz, and A. M. Kosevich, Theory of Elasticity. Butterworth-Heinemann, 1986.

[4] D. L. Russell and L. White, "An elementary nonlinear beam theory with finite buckling deformation properties," SIAM Journal of Applied Mathematics, pp. 1394–1413, 2002.

[5] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, and J. Piater, "Learning Grasp Affordance Densities," Paladyn Journal of Behavioral Robotics, vol. 2, pp. 1–17, 2011.

[6] B. W. Silverman, Density Estimation for Statistics and Data Analysis. Chapman and Hall/CRC, 1986.

[7] M. Bell, "Flexible object manipulation," Ph.D. dissertation, Dartmouth College Hanover, New Hampshire, 2010.

[8] A. Howard and G. Bekey, "Recursive learning for deformable object manipulation," in Advanced Robotics, 1997. ICAR'97. Proceedings., 8th Int.l Conf. on, 1999.

[9] G. Foresti and F. Pellegrino, "Automatic visual recognition of deformable objects for grasping and manipulation," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 34, no. 3, pp. 325–333, aug. 2004.

[10] J. Acker and D. Henrich, "Manipulation of Deformable Linear Objects: From Geometric Model Towards Program Generation," in International Conference on Robotics and Automation (ICRA), 2005.

[11] C. Elbrechter, R. Haschke, and H. Ritter, "Folding paper with anthropomorphic robot hands using real-time physics-based modeling," in Humanoids, 2012.

[12] A. Jordt and R. Koch, "Fast tracking of deformable objects in depth and colour video," in Proceedings of the British Machine Vision Conference, BMVC 2011, S. McKenna, J. Hoey, and M. Trucco, Eds. British Machine Vision Association, 2011.

[13] A. Jordt and R. Koch, "Direct model-based tracking of 3D object deformations in depth and color video," International Journal of Computer Vision, vol. 102, pp. 239–255, 2013, 10.1007/s11263-012-0572-1.

[14] A. Jordt, I. Schiller, J. Bruenger, and R. Koch, "High-resolution object deformation reconstruction with active range camera," in Pattern Recognition. Springer Berlin / Heidelberg, 2010.

[15] A. R. Fugl, H. G. Petersen, and M. Willatzen, "Simulation of Flexible Objects in Robotics," in Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), 2012.

[16] A. R. Fugl, A. Jordt, H. G. Petersen, M. Willatzen, and R. Koch, "Estimation of Material Properties and Pose for Deformable Objects from Depth and Color Images," in Joint Pattern Recognition Symposium (DAGM-OAGM), 2012.

[17] L. Bodenhagen, A. R. Fugl, M. Willatzen, H. G. Petersen, and N. Krüger, "Learning Peg-In-Hole Actions with Flexible Objects," in 4th Int. Conf. on Agents and Artificial Intelligence - Special Session for Intelligent Robotics, 2012.

[18] D. Nehab, "Advances in 3D shape acquisition," Ph.D. dissertation, Princeton University, Sep. 2007.

[19] A. Griesser, T. P. Koninckx, and L. V. Gool, "Adaptive Real-Time 3D Acquisition and Contour Tracking within a Multiple Structured Light System," in Proceedings of the Computer Graphics and Applications, 12th Pacific Conference, ser. PG '04, 2004.

[20] B. Rosenhahn, U. Kersting, K. Powell, R. Klette, G. Klette, and H.-P. Seidel, "A system for articulated tracking incorporating a clothing model," Machine Vision and Applications, vol. 18, no. 1, pp. 25–40, 2007.

[21] C. Cagniart, E. Boyer, and S. Ilic, "Probabilistic Deformable Surface Tracking From Multiple Videos," in ECCV, 2010.

[22] S. Shen, Y. Zheng, and Y. Liu, "Deformable surface stereo tracking-by-detection using second order cone programming," in ICPR. IEEE, 2008, pp. 1–4.

[23] M. Salzmann, V. Lepetit, and P. Fua, "Deformable surface tracking ambiguities," in CVPR. IEEE Computer Society, 2007.

[24] J. D. Schulman, A. Lee, J. Ho, and P. Abbeel, "Tracking deformable objects with point clouds," in proceedings of the International Conference on Robotics and Automation (ICRA), 2013, pp. 1130–1137.

[25] C. Elbrechter, R. Haschke, and H. Ritter, "Bi-manual robotic paper manipulation based on real-time marker tracking and physical modelling," in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 1427 – 1432.

[26] L. Piegl, "On nurbs: a survey," IEEE Comput. Graph. Appl., vol. 11, pp. 55–71, 1991.

[27] R. Erkamp, P. Wiggins, A. Skovoroda, S. Emelianov, and M. ODonnell, "Measuring the elastic modulus of small tissue samples," Ultrasonic Imaging, vol. 20, pp. 17–28, 1998.

[28] I. Meththananda, S. Parker, M. Patel, and M. Braden, "The relationship between shore hardness of elastomeric dental materials and young's modulus," Dental Materials, vol. 25, no. 8, pp. 956–959, 2009.

[29] E. J. Chen, J. Novakofski, W. K. Jenkins, and W. D. O. Jr, "Youngs Modulus Measurements of Soft Tissues with Application to Elasticity Imaging," IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control, vol. 43, no. 1, Jan. 1996.
[30] A. Howard and G. Bekey, "Intelligent learning for deformable object manipulation," in Computational Intelligence in Robotics and Automation, 1999, pp. 15 –20.
[31] B. Frank, R. Schmedding, C. Stachniss, M. Teschner, and W. Burgard, "Learning the elasticity parameters of deformable objects with a manipulation robot," in Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS), 2010.
[32] M. Willatzen and L. Wang, "Mathematical modelling of one-dimensional piezoelectric transducers based on monoclinic crystals," Acta Acustica united with Acustica, vol. 93, pp. 716–721(6), 2007.
[33] J. Mosegaard, "Cardiac surgery simulation - graphics hardware meets congenital heart disease," Ph.D. dissertation, Department of Computer Science, University of Aarhus, Denmark, 2006.
[34] D. Reznik and C. Laugier, "Dynamic simulation and virtual control of a deformable fingertip," in IEEE Int. Conf. on Robotics and Automation, 1996., vol. 2. IEEE, 1996, pp. 1669–1674.
[35] O. Kroemer, R. Detry, J. Piater, and J. Peters, "Combining Active Learning and Reactive Control for Robot Grasping," Robotics and Autonomous Systems, vol. 58, no. 9, pp. 1105–1116, 9 2010.
[36] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in In IEEE International Conference on Robotics and Automation, 2002, pp. 1398–1403.
[37] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," Robotics, IEEE Transactions on, vol. 26, no. 5, pp. 800 –815, oct. 2010.
[38] P. Jiménez, "Survey on model-based manipulation planning of deformable objects," Robotics and Computer-Integrated Manufacturing, vol. 28, no. 2, pp. 154–163, April 2012.
[39] I. Schiller, C. Beder, and R. Koch, "Calibration of a PMD camera using a planar calibration object together with a multi-camera setup," in The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XXXVII. Part B3a, 2008.
[40] N. Hansen, "The CMA evolution strategy: a comparing review," in Towards a new evolutionary computation. Advances on estimation of distribution algorithms. Springer, 2006.
[41] S. M. Han, H. Benaroya, and T. Wei, "Dynamics of Transversely Vibrating Beams Using Four Engineering Theories," Journal of Sound and Vibration, vol. 225, pp. 935–988, 1999.
[42] A. R. Fugl, "Modeling and Simulation of Grasping of Deformable Objects," Ph.D. dissertation, University of Southern Denmark, 2012.
[43] L.-P. Ellekilde and J. A. Jorgensen, "RobWork: A Flexible Toolbox for Robotics Research and Education," Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK), pp. 1 –7, june 2010.
[44] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," Mathematical Programming, vol. 106, no. 1, pp. 25–57, 2006.
[45] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent, "A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling," SIAM Journal on Matrix Analysis and Applications, vol. 23, no. 1, pp. 15–41, 2001.
[46] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet, "Hybrid scheduling for the parallel solution of linear systems," Parallel Computing, vol. 32, no. 2, pp. 136–156, 2006.
[47] L. Piegl and W. Tiller, The NURBS book, 2nd ed. New York, NY, USA: Springer-Verlag New York, Inc., 1997.
[48] M. A. Carreira-Perpinan, "Mode-finding for mixtures of Gaussian distributions," in IEEE Trans. on Pattern Analysis and Machine Intelligence, 2000, pp. 1318–1323.
[49] "Smac moving coil actuators." [Online]. Available: http://www.smac-mca.com

Leon Bodenhagen received the Master degree within Artificial Intelligence in 2009 and the Ph.D. degree in Robotics in 2013, both at Mæ[rsk McKinney Møller Institute at the University of Southen Denmark. Since 2013 he has post doc position at the Mærsk McKinney Møller Institute.

His main research interests cover learning of robotic manipulation actions in industrial contexts and robotic interactions within domestic environments.

Andreas R. Fugl received his Master degree in 2008 and the Ph.D. degree in 2013 both in the area of Robotics at the University of Southern Denmark.

His research interest interests include solid mechanics, numerical algorithms and parallel computation. Andreas Fugl is currently employed at the Danish Technological Institute where he is working with the next generation of robots for industry

Andreas Jordt received his Diploma in computer science in 2009 at Kiel University. He is currently working at the Multimedia Information Processing Group at Kiel University as a Ph.D. student.

His research interests cover robotics and computer vision, including deformation reconstruction and object tracking using color and depth sensors with a focus on real-time processing.

Morten Willatzen is Deputy Director and Professor at the Department of Photonics Engineering, Technical University of Denmark. Having received his PhD from the Niels Bohr Institute at the University of Copenhagen, he held positions at Aarhus University, Max-Planck-Institute for Solid State Research, Germany, and Senior Scientist at Danfoss A/S, DK. In 2000 he became Associate Professor and 2004-2012 Full Professor in Mathematical Modelling at the Mads Clausen Institute, University of Southern Denmark. Morten Willatzen's research interests include solid state physics, metamaterials, flow acoustics, and modelling of thermo-fluid systems.

Knud A. Andersen has since 1982 performed advanced mechanical engineering, development and design of handling equipment and special machinery amongst other many years in the food industry. From 2008 to 2012 he was a Project Manager at the Danish Technological Institute, Robottechnology division.

Martin M. Olsen received the Ph.D. from University of Southern Denmark in 2001 and worked with offline programmed welding robots at the Odense Steel Shipyard from 2001 to 2007. Since 2007 Martin Olsen is employed at the Danish Technological Institute where he is working with the development of the next generation of robots for industry.

Reinhard Koch holds a Diploma and PhD (Dr.-Ing., 1996) in Electrical Engineering from the University of Hannover, Germany. From 1996 - 99 he lead the 3D modeling team at the KU Leuven, Belgium, in the vision group of prof. Luc van Gool. Since 1999 he is head of the Multimedia Information Processing group at the Department of Computer Science of the Christian-Albrechts-University of Kiel.

The research interests of Reinhard Koch are 3D modeling from video, images, and depth sensors, fast camera calibration and tracking algorithms, and the confluence of Computer Graphics and Computer Vision in the field of Mixed and Augmented Reality. He teaches courses in Computer Graphics, Multimedia Technology and Compression, and Image-based 3D Reconstruction.

Henrik G. Petersen is Professor at the Maersk Mc-Kinney Moller Institute, University of Southern Denmark. He received his PhD degree from the Institute for Mathematics and Computer Science, Odense University in 1989 and was then was employed as post. doc. and assistant professor at the same institute. In 1996 he was employed at the Maersk McKinney Moller Institute, where he became professor in 2002.

His research interests are mainly within mathematical modeling of robots and robotic processes and how to use these models for advanced robot programming and control.

Norbert Krüger is a professor at the Mærsk McKinney Møller Institute, University of Southern Denmark. He holds a M.Sc. degree from the Ruhr-Universität Bochum, Germany and his Ph.D. degree from the University of Bielefeld, Germany. His research covers computer vision, cognitive systems and applied robotics. A particular focus of his work is the use of cognitive methodologies in robot applications in an industrial and welfare context.